XP & Scrum

Beatrice Åkerblom
beatrice@dsv.su.se

XP Roles

- **Customer**
  - Writes User Stories and specifies Functional Tests
  - Sets priorities, explains stories
  - May or may not be an end-user
  - Has authority to decide questions about the stories

- **Programmer**
  - Estimates stories
  - Defines Tasks from stories, and estimates
  - Implements Stories and Unit Tests

XP Roles, cont’d

- **Coach**
  - Watches everything, makes sure the project stays on course
  - Helps with anything

- **Tracker**
  - Monitors Programmers’ progress, takes action if things seem to be going off track.
  - Actions include setting up a meeting with Customer, asking Coach or another Programmer to help
XP Roles, cont’d

- **Tester**
  - Implements and runs Functional Tests (not Unit Tests!)
  - Graphs results, and makes sure people know when test results decline.

- **Doomsayer**
  - Ensures that everybody knows the risks involved
  - Ensures that bad news isn't hidden, glossed over, or blown out of proportion

Manager

- Schedules meetings (e.g. Iteration Plan, Release Plan), makes sure the meeting process is followed, records results of meeting for future reporting, and passes to the Tracker
- Possibly responsible to the Gold Owner.
- Goes to meetings, brings back useful information
- Pays for pizza

Gold Owner

- The person funding the project, which may or may not be the same as the Customer

XP Practices

XP is based on 12 key practices:

- The Planning Process
  - Release Planning & Iteration Planning
- Frequent, Small Releases
- System Metaphor
- Simple Design
- Test Driven Development
- Refactoring
- Pair Programming
- Collective Code Ownership
- Continuous Integration
- Sustainable Pace
- On-site Customer
- Coding Standard

Stages of an XP project

- **Initiation**
  - User Stories
- **Release Planning**
- **Release** (each Release is typically 1 -6 months)
  - Iteration 1 (typically 1 -3 weeks)
    - Development
    - Deployment
    - Acceptance Testing
  - Iteration 2
    - Development
    - Deployment
    - Acceptance Testing
  - ...

5

6

7

8
**Gathering Requirements**

- **Responsibilities**
  - Key Point: The Customer is responsible for the requirements.
  - Programmers help to gather and clarify requirements. Customers especially need help with non-functional requirements and with working out the details of acceptance tests.

- **Documentation**
  - User Stories
  - Acceptance Test Cases

**User Stories cont’d**

- User stories have three crucial aspects:
  - **Card**
    - Enough information to identify the story
  - **Conversation**
    - Customer and Programmers discuss the story to elaborate on the details
    - Verbal when possible, but documented when required
  - **Confirmation**
    - Acceptance tests to confirm that the story has been properly implemented
User Stories

- A short description of the behaviour of the system from the point of view of the Customer
- Use the Customer’s terminology without technical jargon
- One for each major feature in the system
- Must be written by the users
- Are used to create time estimates for release planning
- Replace a large Requirements Document

User Stories cont’d

- Drive the creation of the acceptance tests:
  - Must be one or more tests to verify that a story has been properly implemented
- Different than Requirements:
  - Should only provide enough detail to make a reasonably low risk estimate of how long the story will take to implement.
- Different than Use Cases:
  - Written by the Customer, not the Programmers, using the Customer’s terminology
  - More “friendly” than formal Use Cases

Acceptance Tests

- Formal test to determine if a system satisfies its acceptance criteria, i.e. the User Stories!
- Should be automated, but may simply be a series of repeatable steps
- At least one Acceptance Test for each Story

Release Planning

- Customer defines the business value of desired features (User Stories)
- Programmers provide estimates of 1, 2 or 3 “points”
- Stories larger than 3 points must be split into smaller stories
- Customer decides which Stories are to be included in a Release
- Focus on completing the Stories with the highest business value and highest risk first
Release Planning cont’d

- Stories for a Release are arranged into 1-3 week Iterations
- Higher risk, and higher priority stories in earlier Iterations
- For new system, the 0:th Iteration defines the basic skeleton of the application and infrastructure required
- The Release and Iterations have fixed dates for completion – dates are fixed, scope is variable
- This is the Release Plan

Iteration Planning

- Stories for Iteration are broken down into Tasks by Programmers
- Tasks are estimated by all Programmers as a group
- Programmers “sign up” for Tasks, and estimate the time to complete
- Can only sign up for as many points as were completed in the last Iteration
- Once development begins, Project Velocity measures progress

Programmer Tests

- Automated tests written to test the behaviour of individual classes
- Fundamental to XP, and maintaining a flat cost curve
- XP uses a Test First mentality; write the test, then write to code to make the test pass.
  - "Never write a line of code without a failing test." (Kent Beck)
- No code goes into production unless it has associated tests
- Tests are written first
- Tests determine what code you need to write

Programmer Tests cont’d

- Programmer Tests must run at 100% before code is integrated
- At most one test failing at any time
- “Grey-box” testing
- Assist with Refactoring (promote Courage)
- Testing frameworks exist for many languages:
  - JUnit for Java
  - CPPUnit for C++
  - NUnit for all .Net languages
Small Releases

- Start with the smallest useful feature set
- Release early and often, adding a few features each time
- Each iteration ends in a release

System Metaphor

- Each project has an organizing metaphor, which provides an easy to remember naming convention
- The names should be derived from the vocabulary of the problem and solution domains

Simple Design

- Always use the simplest possible design that gets the job done
- The requirements will change tomorrow, so only do what’s needed to meet today’s requirements
- Uses the fewest number of classes and methods

Refactoring

- Refactor out any duplicate code generated in a coding session
- You can do this with confidence that you didn’t break anything because you have the tests
- “Refactoring-Improving the Design of Existing Code”, by M. Fowler, 1999 Addison-Wesley
Pair Programming

- All production code is written by two programmers sitting at one machine
- Essentially, all code is reviewed as it is written
- Helm – keyboard and mouse doing implementation
- Tactician – Thinking about the implications and possible problems

Experiences using Pair Programming

- Reported productivity person month [R. Jensen]
  - Single programmer 77 source lines (historical base line)
  - Pair programming 175 source lines
- Cockburn & Williams
  - Development costs are an additional 15%
  - Resulting code has about 15% fewer defects

Remember this?

Collective Code Ownership

- No single person "owns" a module.
- Any developer is expect to be able to work on any part of the code base at any time
- Improvement of existing code can happen at anytime by any pair
Continuous Integration

- All changes are integrated into the code base at least daily
- The tests have to run 100% both before and after integration

40-Hour Work Week

- Programmers go home on time. In crunch mode, up to one week of overtime is allowed
- Multiple consecutive weeks of overtime are treated as a sign that something is very wrong with the process

On-site Customer

- Development team has continuous access to a real live customer, that is, someone who will actually be using the system
- For commercial software with lots of customers, a customer proxy (usually the product manager) is used instead

Coding Standards

- Everyone codes to the same standards
- Ideally, you shouldn't be able to tell by looking at it who on the team has touched a specific piece of code.
Scalability (Team Size)

- XP works well with teams up to 12-15 developers
- It tends to degrade with teams sizes past 20
- Work has been done in splitting large projects/teams into smaller groups and applying XP within each group

Environment

- Programmers must be located physically close, often in the same room and desk
- Iterations typically last 1-3 weeks
- Teams will typically use the same duration for all iterations
- Tests are written before the code is written
- End of iteration delivers a working system

eXtreme?

Taking proven practices to the extreme:

- If testing is good, let everybody test all the time
- If code reviews are good, review all the time
- If design is good, refactor all the time
- If integration testing is good, integrate all the time
- If simplicity is good, do the simplest thing that could possibly work
- If short iterations are good, make them really, really short

Literature
Reading For Today:

- Cockburn’s book chapter 4 & 5
  - What is a methodology? What is the difference between a methodology and “the way we usually do things in our work team”?
  - Why methodology?

Scrum -- Introduction

- SCRUM is a loose set of guidelines that govern the development process of a product, from its design stages to its completion.
- SCRUM has been successfully employed by hundreds of different companies in many different fields, with outstanding results.
- There are many similarities between SCRUM and XP, but one of the major differences is that SCRUM is a fairly general set of guidelines that govern the development process of a product.
  - SCRUM is often used as a "wrapper" for other methodologies, such as XP or CMM, to guide the overall process of development when using these other methodologies.

SCRUM Values

- The SCRUM values are derived from the Agile values of software development:
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan
The SCRUM Process

- The scrum process has 3 main phases:

<table>
<thead>
<tr>
<th>Pre-game</th>
<th>Mid-game</th>
<th>Post-game</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning and high-level design</td>
<td>Develop</td>
<td>Wrap</td>
</tr>
<tr>
<td>Adjust</td>
<td>Review</td>
<td>Closure</td>
</tr>
</tbody>
</table>

Sprint Cycle

Daily SCRUM Meeting

- A 15-minute SCRUM meeting is held every day.
- The SCRUM Master asks the three questions, and all members of the team and interested parties take part and give feedback.
- The meeting should be held at the same place every time, so that people know where to go.

1. What have you accomplished since the last meeting?
2. Are there any obstacles in the way of meeting your goal?
3. What will you accomplish before the next meeting?

Sprint

- Before a sprint is begun, a Sprint Planning Meeting is held to determine what features are to be implemented in that sprint.
  - Develop the product further - implement, test, and document.
  - Wrap up the work - get it ready to be evaluated and integrated.
  - Review the work done in this sprint.
  - Adjust for any changes in requirements or plans.
Creating A Backlog

- The product owner compiles all the requests and specifications that are the basis of the changes of the product, e.g. new functions and bug fixes. After the goals have been defined, the entirety is broken down into segments
- Each segment should in part create business value and in part be sub-deliverable
- A prioritised list is made at the same time – the product owner personally makes the decisions at this point
- When it is time to start a new Sprint, the product owner “freezes” the foremost items on the to-do list and summons the SCRUM team to a meeting

Backlog

1. Product Backlog -- a repository for requirements targeted for release at some point. High level requirements with high level estimates provided by the product stake-holders
2. Release Backlog - Requirements pulled from the product backlog and identified and prioritised for an upcoming release. Contains more details about the requirement and low level estimate which are usually estimated by the team
3. Sprint Backlog - A result from each sprint planning is a backlog of requirements/sub-requirements estimated to be completed at the end of the sprint where the requirements from the release backlog are broken down into manageable chunks that can be accomplished typically in 8 - 16 hours.

The Scrum Team

- The SCRUM team consists of 2 groups:
  - the interested team, which consists of people who are interested, but who will not be doing the work
  - the working team -- people who are interested, and will be doing the work on the project

The Scrum Team, cont’d

- A SCRUM is a self-empowered team where everyone had the global view of the product on a daily basis
- The development team should perform as a sport team, every team member working independently but towards the same goal
- A team typically 6-9 working members, although SCRUM has been successfully used with more members
- The team members decide how the work is arranged and how assignments are distributed
- There are no set project roles – everyone should be able to swap tasks with another member
The Scrum Team, cont’d

- If a project is run in more than one SCRUM team, there should be people to act as bridges between the teams
  - attending meetings of more than one SCRUM team
  - act as a communication bridge between the teams

The team leader (SCRUM Master)

- The team's leader is called the SCRUM Master
- The SCRUM Master should be one of the members of the working team -- that is, he should be one of the people who is actually doing the work on the project
- The SCRUM Master measures progress, removes impediments, and leads the team meetings

The Product Owner

- The product owner represents the voice of the customer and ensures that the Scrum team works with the right things from a business perspective
- The product owner administers a product Back-log
- The product owner is often a customer, but can also be part of the internal organisation

SCRUM divided into 3 types by Jeff Sutherland

- Type A - Isolated cycles of work
- Type B - Overlapping iterations
- Type C - All at once
Tyranny of the Sprint

End of Today’s Lecture

Thanks for your attention!